UNIVERSITETET I OSLO

Det matematisk-naturvitenskapelige fakultet

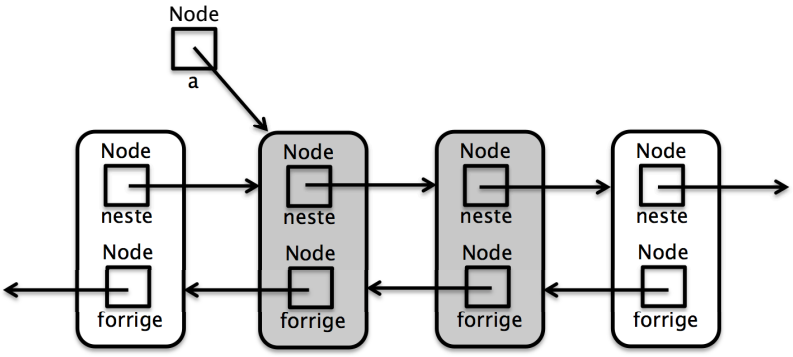
Eksamen i	INF1010 — Objektorientert programmering
Dato:	9. juni 2016
Tid for eksamen:	09.00 – 15.00 (6 timer)
Oppgavesettet er på 7 sider.	
Vedlegg:	Kapittel 20 fra BIG JAVA
Tillatte hjelpemidler:	Læreboka (med notater i)

Kontroller at oppgavesettet er komplett før
du begynner å besvare spørsmålene.

Les gjennom hele oppgaveteksten før du begynner å svare. Noter ned uklarheter/spørsmål under gjennomlesningen slik at du har spørsmålene klare når faglærer kommer. Husk å skrive i besvarelsen der du gjør egne forutsetninger. Forklar med ord hva en kode det ikke eksplisitt er spurt etter, gjør.

Dobbel lenkeliste—datastruktur

Figuren nedenfor er et utsnitt av datastrukturen i ei lenkeliste.



Oppgave 1 — 3%

a) Hvor mange objekter er det i figuren?

b) Hvor mange forskjellige typer er det?

c) Hvor mange variabler er det i figuren?

Oppgave 2 — 4%

Skriv metoden `void bytt(Node n)` som etter kallet `bytt(a)` har byttet om noden som `a` peker på med noden `a.neste`. På figuren er det de to midterste nodeobjektene som er farget litt gråere. Du kan anta at det er en node før og en etter de to nodene som skal byttes om.

(Fortsettes på side 2.)

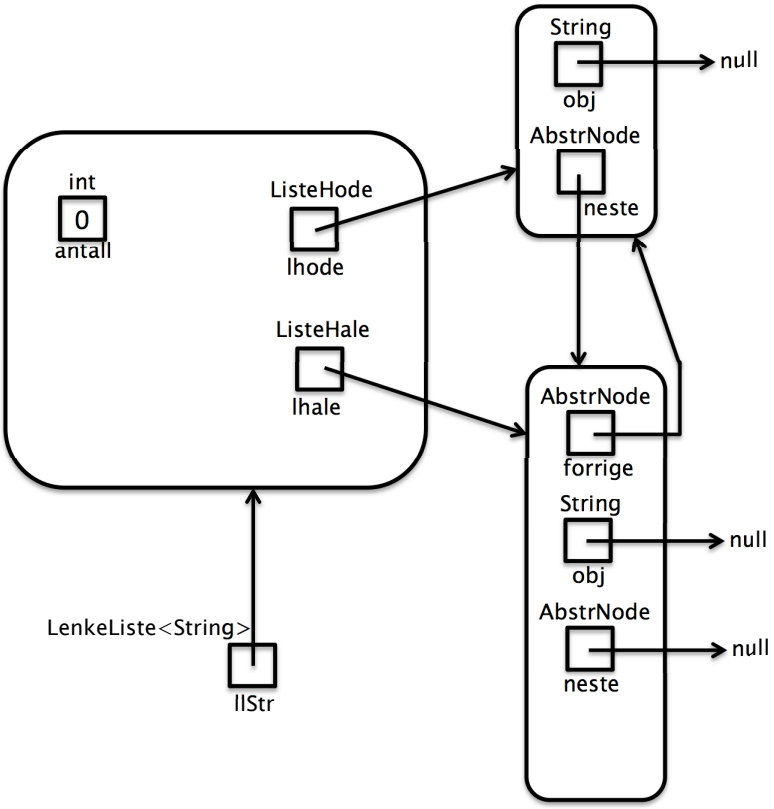
Lenkelistebeholder

Her er en programskisse for en beholder som bruker ei lenkeliste som datastruktur.

```
public class LenkeListe< T extends Comparable<T> > {
    private ListeHode lhode;   // listehodet
    private ListeHale lhale;   // listehalen
    private int antall;        // antall objekter i lista

    LenkeListe() {  }

    private abstract class AbstrNode {
        T obj;
        AbstrNode neste;

        AbstrNode(T t) {
            obj = t;
            neste = null;
        }

        abstract int compareTo(AbstrNode k);
        abstract void settInnOrdnet(AbstrNode k);
    }

    private class ListeHode extends AbstrNode { }
    private class ListeHale extends AbstrNode { }
    private class Node extends AbstrNode {
        ...
        int compareTo(AbstrNode k) {
            return obj.compareTo(k.obj);
        }
        ...
    }

    public int antall() { return antall; }

    // denne metoden skal ikke endres
    public void settInnOrdnet (T nyComparable){
        Node nyNode = new Node(nyComparable);
        lhode.settInnOrdnet(nyNode);
    }

    public void settInnBak (T nyComparable) { }
    public ...  taUtForan () { }
    public boolean tom() { }

}
```

Lenkelista har et listehode og en listehale. Her er datastrukturen som skal oppstå når vi lager en ny beholder:

(Fortsettes på side 3.)



Oppgave 3 — 5%

Skriv klassene LenkeListe, Listehode, ListeHale og Node med klassevariable og konstuktører slik at tilstanden i beholderen blir slik som figuren viser når vi nettopp har laget en tom beholder, jf. main-metoden i oppg. 9.

Oppgave 4 — 2%

Legg merke til hvordan metoden compareTo() er definert i Node og at vi i denne beholderen har definert denne metoden til å sammenligne noder.

Implementer metoden compareTo() i ListeHode.

Oppgave 5 — 1%

Implementer metoden compareTo() i ListeHale.

Oppgave 6 — 3%

Invariant tilstandspåstand: *Variabelen* forrige *i listehalen skal hele tida peke på bakerste node. Den skal peke til listehodet når beholderen er tom.* Denne påstanden skal gjelde initielt og etter metodekall som endrer på datastrukturen i lenkelista.

Implementer metoden settInnBak i beholderen slik at det nye objektet legges bakerst. Denne metoden skal ikke være rekursiv.

Oppgave 7 — 10%

Legg merke til hvordan settInnOrdnet er definert i beholderen. Den kaller på den rekursive settinn-metoden i listehodet. Skriv ferdig klassene ListeHode,

(Fortsettes på side 4.)

ListeHale og Node ved å implementere den *rekursive* metoden settInnOrdnet der den er påkrevd. Gjør det tydelig i svaret hvilken metode som hører til hvilken klasse.

Oppgave 8 — 4%

Programmer metoden taUtForan i klassen LenkeListe slik at første node (ikke listehodet) fjernes og objektet som obj peker på returneres til kallstedet. Hvis lista er tom skal metoden kaste et unntak.

Oppgave 9 — 5%

```
class Main  {
    public static void main (String[] a) {
      LenkeListe<String> llStr =  new LenkeListe<String>();
      // datastrukturen her i datastrukturtegninga ovenfor
      llStr.settInnBak(new String("sss"));
      llStr.settInnOrdnet(new String("yyy"));
      llStr.settInnOrdnet(new String("ttt"));
      llStr.settInnBak(new String("aaa"));
      llStr.settInnOrdnet(new String("bbb"));
    }
}
```

Tegn datastrukturen som blir laget av main-metoden i programskissen ovenfor. I tillegg til objektene som settes inn, skal du tegne listehodet og -halen.

Oppgave 10 — 20%

Siden hverken listehodet eller listehalen trenger obj-variabelen, og listehalen ikke trenger nestepeker, ønsker vi å ta bort disse variablene fra AbstrNode. Vi blir da stående igjen med en klasse som kan defineres som et grensesnitt. La oss kalle dette grensesnittet Elem, siden det er grensesnittet for alle listeelementer (noder og listeender). Skriv grensesnittet og klassene Listehode, ListeHale og Node på nytt. Du trenger ikke skrive metoder som ikke endres.

Sortere tekststrenger

Vi har en fil med 390000 ord (tekststrenger) som vi skal sortere, dvs. ordne i alfabetisk stigende orden. For å avgjøre om et ord er alfabetisk større eller mindre enn et annet ord, skal vi bruke metoden compareTo() fra klassen String.

En skisse av en algoritme for hvordan man kan gå fram for å ende opp med ei lenkeliste hvor alle ordene ligger ordnet (du skal ikke programmere denne ikke-parallelle algoritmen):

- Først oppretter vi en String [ ] som har plass til 390000 ord. Arrayen fylles fra filen med navn mangeord ved hjelp av den ferdiglagde metoden

  ```
  String [] lesInnOrdFraFil(String filnavn)
  ```

(Fortsettes på side 5.)

- Så oppretter vi en beholder av klassen LenkeListe fra forrige oppgave og legger inn 10000 ord fra arrayen, slik at de blir ordnet (innstikksortering). Til dette brukes metoden settInnOrdnet.

- Dette gjør vi 39 ganger, og hver gang vi har fylt opp en ny beholder med 10000 ord fra arrayen, *flettes* den sammen med en resultatbeholder til en ny resultatbeholder.

  Til slutt står vi igjen med en resultatbeholder med 390000 ord.

Legg merke til at vi kun bruker arrayen med ordene som et mellomlager. Vi bruker beholdere både til innsetting (sortering) og fletting. Til flettingen av to beholdere bruker vi metoden flett, som er delvis ferdiglaget:

```
LenkeListe<String> flett(LenkeListe<String> a, LenkeListe<String> b) {
  // fletter de ordnede listene a og b sammen til ei ny ordnet liste c

    LenkeListe<String> c = new LenkeListe<String>();
    String fraA = a.taUtForan();
    String fraB = b.taUtForan();
    while ( fraA != null && fraB != null ) {
        if ( fraA.compareTo(fraB) <= 0 ) {
            c.settInnBak(fraA);
            fraA = a.taUtForan();
        }
        else {
            c.settInnBak(fraB);
            fraB = b.taUtForan();
        }
    }

    // hva vet vi om listene a og b her ?

    /* her er noe kode utelatt som oppgave (11b) */

}
```

Oppgave 11 — 4%

a) Hva vet vi om de to listene a og b etter while-løkka i metoden flett?

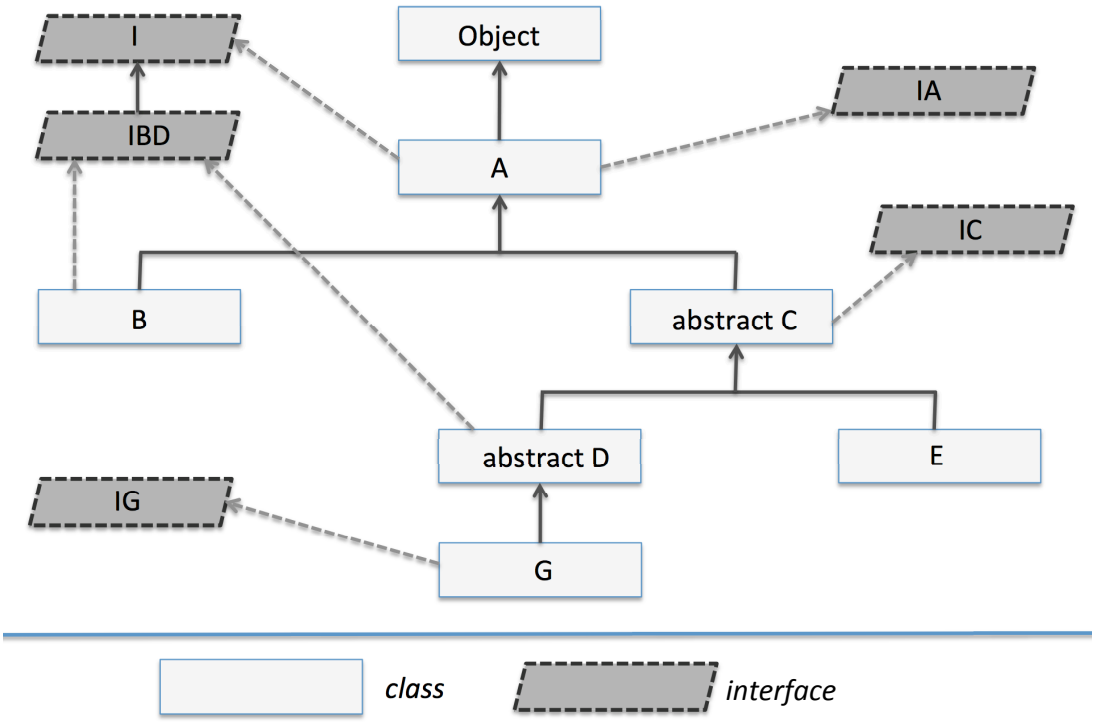
b) Fullfør metoden flett.

Oppgave 12 — 20%

Skriv en main-metode som bruker tråder til å parallellisere innstikksorteringen (innsettingen) og flettingen. Du *kan*, men *må ikke*, bruke wait()/notify i denne oppgaven. Men både *innsettingen* i lenkelistene og *flettingen skal* skje i parallell.

(Fortsettes på side 6.)

Klasser, grensesnitt og arv



Oppgave 13 — 5%

Skriv klasse- og interface-definisjoner som svarer til hierarkiet ovenfor. Definisjonene skal være tomme, dvs det ikke skal stå noe mellom { og }. Det skal ikke være andre klasser eller grensesnitt enn dem i figuren. Skriv alle grensesnitt først, etterfulgt av de abstrakte klassene. Tilslutt ikke-abstrakte klasser i alfabetisk rekkefølge. En definisjon pr. linje.

Oppgave 14 — 4%

a) Hvor mange forskjellige objekter kan man opprette?

b) Hvor mange metoder er det mulig å kalle i et objekt av klassen B når vi legger til grunn at Object har 11 metoder, at klassene ikke har egne metoder og at alle interfacene i hierarkiet/programmet har 2 metoder hver, alle med forskjellig signatur?

c) Hvor mange metoder er tilgjengelig i et objekt av klassen G?

d) Hvor mange metoder er det mulig å kalle fra variabelen ia etter tilordningen på linje 8 i programmet nedenfor?

(Fortsettes på side 7.)

Vi skriver definsjonen av klassen Mainklasse sammen med definisjonene fra oppgave 13:

```
1  public class Mainklasse {
2      public static void main(String [] args) {
3
4          C c = new E();
5          IC ic = c;
6          IBD ibd = c;
7          Object o = new B();
8          IA ia = (IA) o;
9          ic = (IC) o;
10         A a = (C) ic;
11         E e = new A();
12         IBD ibd = (IBD) a;
13         a = (B) o;
14         c = new C();
15
16     }
17 }
```

Oppgave 15 — 6%

Fjern setningene som feiler enten under kompilering eller ved kjøring. Skriv main-metoden på nytt uten setningene som feiler.

Oppgave 16 — 4%

Tegn datastrukturen slik den er rett før main-metoden avslutter. Du trenger ikke tegne variable som har verdien null eller objekter som ingen variabel peker/refererer til.

Oppgavesett slutt.

Lykke til, og god sommer!

Stein Michael Storleer